Чекпоинт 3: Деплой и интерфейс

RAG Чат-бот по роману «Мастер и Маргарита»

Команда bookworm

Горбунов Дмитрий Павлович
Ковалева Дарина Евгеньевна
Тельнов Федор Николаевич
Мацаков Борис Вячеславович

Репозиторий: https://github.com/bookworm-itmo/llm-intro-project

1. Функционал интерфейса

Веб-интерфейс реализован на **Streamlit** и представляет собой чат-бота для ответов на вопросы по роману М.А. Булгакова «Мастер и Маргарита».

Основные возможности

Функция	Описание
Чат-интерфейс	Классический формат диалога с историей сообщений
RAG-поиск	Поиск релевантных фрагментов через FAISS + GigaChat Embeddings
Генерация ответов	Claude Haiku 4.5 на основе найденного контекста
Источники	Раскрывающийся блок с цитатами из книги под каждым ответом
Реранкер	Переключатель в сайдбаре для BGE Reranker

Элементы интерфейса

1. **Заголовок** — «Чат-бот по роману ‘Мастер и Маргарита’» с пояснением
2. **Сайдбар** — чекбокс «Использовать реранкер» (включен по умолчанию)
3. **История чата** — все предыдущие вопросы и ответы сессии
4. **Поле ввода** — внизу экрана, placeholder «Задайте вопрос о книге»
5. **Спиннер** — индикатор загрузки «Ищу ответ в книге...»
6. **Экспандер источников** — «Источники из книги» с номером главы и текстом

Обработка ошибок

- При отсутствии API-ключей — ошибка при запуске
- При сбое LLM/embedding API — исключение отображается в интерфейсе
- Rate limit — обрабатывается с retry и задержками

2. Скриншот интерфейса

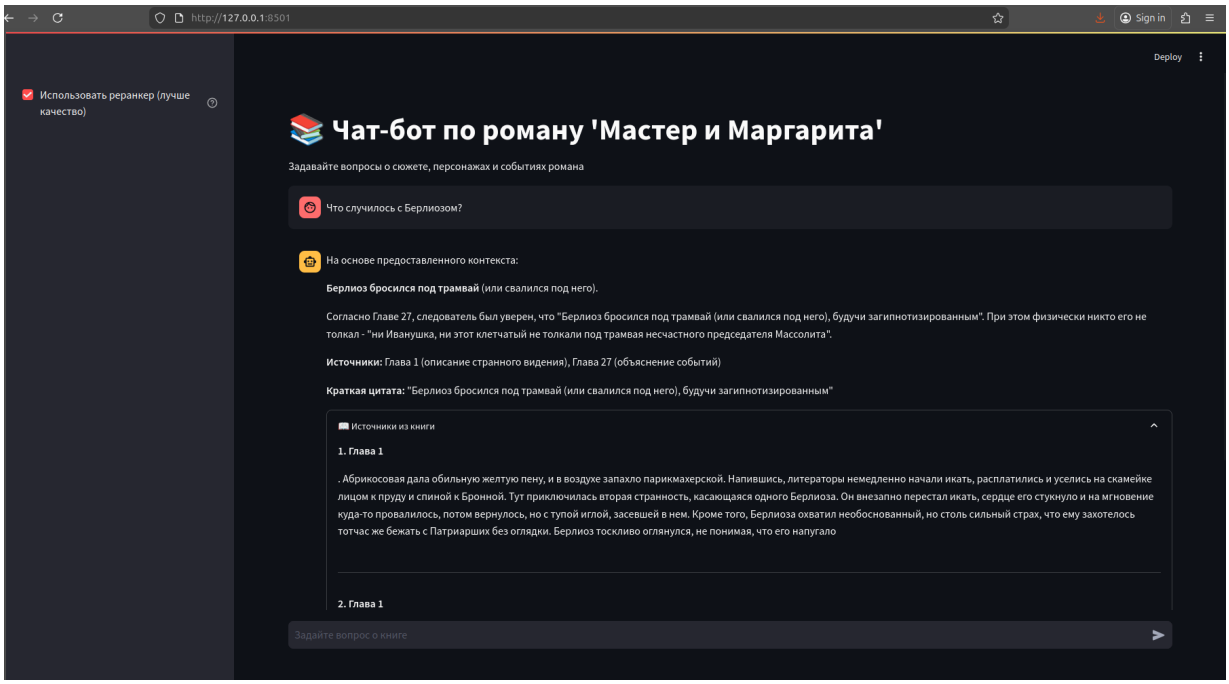

Рис. 1: Интерфейс чат-бота: вопрос «Что случилось с Берлиозом?», ответ с источниками, развернутый блок цитат из глав. Слева — сайдбар с переключателем реранкера.

3. Инструкция по запуску

Требования

- Python 3.10+
- API-ключи Claude (Anthropic) и GigaChat (Сбер)

Быстрый старт

```
# Клонировать репозиторий
git clone https://github.com/bookworm-itmo/llm-intro-project
cd llm-intro-project

# Установить зависимости
pip install -r requirements.txt

# Создать .env файл
echo "CLAUDE_API_KEY=your_key" > .env
echo "GIGACHAT_AUTH_KEY=your_key" >> .env

# Подготовить данные (если нет готовых)
python main.py
```

```
# Запустить интерфейс
streamlit run frontend/app.py
```

Запуск через Docker

```
docker-compose up frontend
```

Структура .env

```
CLAUDE_API_KEY=sk-ant-...
GIGACHAT_AUTH_KEY=base64_encoded_key
```

4. Финальные метрики качества системы

4.1 Метрики Retrieval (поиск релевантных фрагментов)

Сравнение с реранкером и без:

Метрика	Без реранкера	С реранкером	Изменение
Precision	0.373	0.453	+21.4%
Recall	0.528	0.550	+4.2%
F1	**0.404**	**0.491**	**+21.5%**

Таблица 1: Сравнение качества retrieval с реранкером и без

Реранкер: BGE-reranker-base с гибридной стратегией (weight=0.7, multiplier=3.0x)

4.2 Метрики RAGAS (качество генерации)

Метрика	Среднее	Медиана	Описание
Faithfulness	0.755	0.764	Соответствие ответа контексту
Answer Relevancy	0.418	0.000	Релевантность ответа вопросу
Context Recall	0.642	1.000	Полнота найденного контекста
Context Precision	0.360	0.287	Точность контекста
Context Utilization	0.382	0.287	Использование контекста моделью

Таблица 2: Метрики RAGAS на валидационной выборке (70 запросов)

4.3 Анализ результатов

Сильные стороны:

- Высокий Faithfulness (0.755) — модель редко галлюцинирует

- Context Recall медиана 1.0 — в большинстве случаев retrieval находит информацию
- Реранкер улучшает F1 на 22%

Слабые стороны:

- Answer Relevancy медиана 0.0 — модель часто отказывается отвечать
- 44.3% отказов («недостаточно информации в контексте»)
- 15 случаев отказа при полном context_recall=1.0

5. Проведенные эксперименты

5.1 Оптимизация реранкера

Протестировано **105 комбинаций** гиперпараметров:

- Вес реранкера: 0.5–0.85
- Множитель кандидатов: 2.5x–4.5x
- Стратегии комбинирования: linear, rrf, geometric

Лучшая конфигурация:

```
reranker_weight = 0.7
candidate_multiplier = 3.0
score_combination = "linear"
```

5.2 Что работало

- **Реранкер** — значительное улучшение precision (+21%)
- **Гибридная стратегия скоринга** — лучше чем чистый semantic search
- **GigaChat Embeddings** — хорошо работают для русского текста
- **Claude Haiku** — быстрые и качественные ответы

5.3 Что не работало

- **Консервативный промпт** — слишком много отказов
- **OpenAI Embeddings** — проблемы с VPN и доступом
- **Большой top_k без реранкера** — много нерелевантного шума

6. Архитектура системы

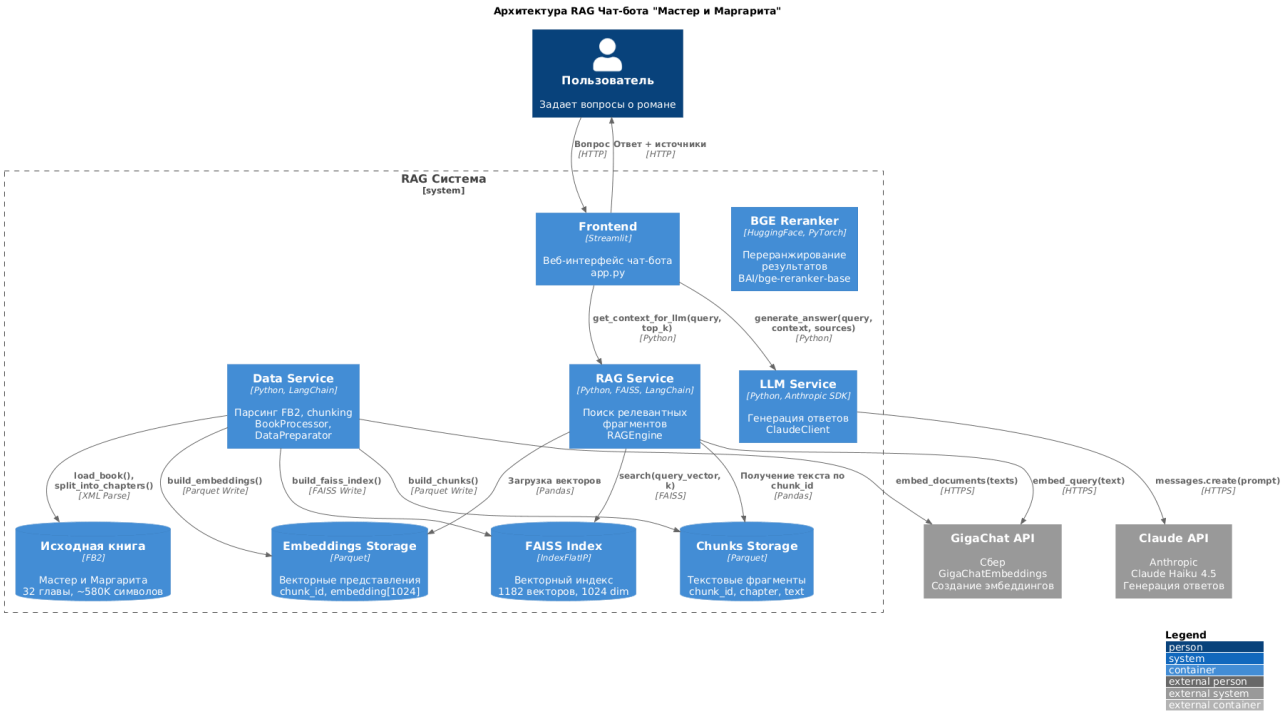

Рис. 2: C4-диаграмма архитектуры RAG чат-бота (PlantUML)

7. Структура репозитория

```
llm-intro-project/
 data/                    # Данные (chunks, embeddings, index)
 services/
    data_service/        # Парсинг FB2, chunking
    rag_service/         # FAISS, embeddings, reranker
    llm_service/         # Claude API
 frontend/                # Streamlit UI
 validation/              # Скрипты оценки
 metrics/                 # Результаты экспериментов
 docs/                    # Документация
 requirements.txt         # Зависимости (точные версии)
 README.md               # Инструкция по запуску
```

Заключение

Реализован полнофункциональный RAG чат-бот с веб-интерфейсом для ответов на вопросы по роману «Мастер и Маргарита». Система использует:

- **GigaChat Embeddings** для векторизации текста (1024 dim)

- **FAISS IndexFlatIP** для быстрого поиска
- **BGE Reranker** для улучшения качества (+22% F1)
- **Claude Haiku 4.5** для генерации ответов
- **Streamlit** для веб-интерфейса

Система может быть развернута локально через pip или Docker. Реранкер включен по умолчанию для лучшего качества ответов.